Facebook Friends

In this assignment, you will use linked lists to implement the "friend" functionality of a Facebook-like application.

You will define a class called "User" that represents people with accounts in your application. You will save all of the User objects in an unordered linked list by adding them to the list as you create each new User.

Each User object will store the user's name (a string). It will also store a list of the user's friends (other User objects) in an unordered linked list.

Commands

When your program begins executing, it should enter a loop where it reads a line from the provided data file, interprets the line as a command, prints out the command, and then executes it. There are six possible commands:

- **add an account**: create a new User object with the specified name. The data line in the file will look like:

  ```
     Person name

  Example:
     Person Kanye
  ```

  If the command is valid, your program should print out the message, "*name* now has an account."

- **add a friend**: update your data structure to indicate that two people are now friends. The data line in the file will look like:

  ```
     Friend name1 name2

  Example:
     Friend Swift Sheeran
  ```

  If the command is valid, your program should print out the message, "*name1* and *name2* are now friends."

- **delete a friend**: update your data structure to indicate that two people are no longer friends. The data line in the file will look like:

  ```
     Unfriend name1 name2

  Example:
  ```

```
        Unfriend Angelina Brad
```

If the command is valid, your program should print out the message, "*name1* and *name2* are no longer friends."

- **list all friends**: print out the names of all Users who are currently friends with the specified person. The data line in the file will look like:

```
        List name

    Example:
        List Kanye
```

If the command is valid, your program should print out a list of the names of the person's friends on one line. **(Note: I changed this from "one per line" because it was not consistent with the sample output file. If you already did it one per line, that's okay: I'll accept either.)** If the person has no friends, your program should print out the message, "*name* has no friends."

- **query friend status**: check whether or not two specified Users are currently friends. The data line in the file will look like:

```
        Query name1 name2

    Example:
        Query Donald Hillary
```

If the command is valid, and if the two Users are currently friends, your program should print the message "*name1* and *name2* are friends." If they are not currently friends, print the message, "*name1* and *name2* are not friends."

- **exit the program**: terminate the program. The data line in the file will look like:

```
        Exit

    Example:
        Exit
```

Your program should print out the message, "Exiting..." and then terminate.

Input Validation

Your program does not have to check for bad command structure; you can assume that every command word is spelled correctly, has the correct number of arguments, etc. However, your program should be able to gracefully handle commands that do not make any sense, and print out appropriate messages. At the very least, your program must handle the following circumstances:

- If a Person command attempts to create a User with a name that has already been created, print an appropriate message and continue without changing anything.
- If a command references a User that does not yet have an account, print an appropriate message and continue without changing anything. Note that it should print out two messages if the command includes two invalid user names.
- A User cannot Friend or Unfriend him/herself. Print an appropriate message and continue without changing anything.
- A User cannot Unfriend someone who is not his/her friend. Print an appropriate message and continue without changing anything.
- A User cannot Friend someone who is already his/her friend. Print an appropriate message and continue without changing anything.

Specific Program Requirements:

- You must define a class "User", which has an instance variable to store the User's name, and an instance variable to store the User's friends.
- You must store the collection of User objects in a linked list. You may choose any implementation we've discussed in class, including unordered or ordered, with or without sentinels, singly or doubly-linked, or circular.
- You must store the friends of each User in a linked list. Note that this means friending or unfriending a person affects the friends lists of two User objects. Again, you may choose any implementation we've discussed in class.
- The linked list of friends must be a list of User objects, not a list of strings (in particular, the names of the friends).

Input:

- The data file for this program is FriendData.txt.

Output:

- The output from your program should look something like FriendOutput.txt.